The Development and Software Design of a Cost Effective, Mobile CAVE

Competency:
Division:
August 11, 2004

Manager:

____
Ashley Kramer

Team Members:

____
John Wilkinson

____
Cris Kania

____
Kavi Jivan

Mentors:

____
Janice Cawthorn
DEVELOP Langley Faculty Advisor
Phone: (757) 864-3834
Email: j.d.cawthorn@larc.nasa.gov

NASA Langley Research Center
4 Langley Blvd.
Building 1230, Room 137
Hampton, VA 23681-2199

____
Michael Ruiz
National DEVELOP Program Manager
Phone: (757) 864-3738
Fax: (757) 864-7890
Email: m.l.ruiz@larc.nasa.gov

NASA Langley Research Center
4 Langley Blvd.
Building 1230, Room 133 C
Hampton, VA 23681-2199

Abstract

NASA's DEVELOP program has spent the past three years implementing a portable Cave Automatic Virtual Environment (CAVE). A CAVE is a multi-person 3D immersive environment consisting of three or more walls and based on an active stereo imaging system. A traditional CAVE is primarily used as a research tool, running calculation intense simulations, costing upwards of one million dollars, and lacking the ability to be easily transported. The primary motive for undertaking the development of a portable CAVE was to provide a mobile gizmo for DEVELOP's other Earth Science teams to more effectively present and explain their data visualizations. Our portable CAVE can collapse in under an hour, load easily into a fifteen passenger van, transported anywhere, and can be setup and aligned in under two hours. In the project's first two years, the team designed and implemented the parallel computing system and the CAVE's associated hardware. The portable CAVE was designed using all commercially available Linux-based PC's, standard LCD projectors, and a novel shutter wheel system for the active stereo imaging. This brought the cost down to around $35,000, a mere fraction of the cost of a traditional CAVE. This summer's work has focused on the development of the software to drive the CAVE and import an array of Earth Science data sets. The main file types that this CAVE needed to support were DEMS, overlays, and Geospatial Feature Sets. After much research, we decided to use the Virtual Terrain Project (VTP), Simple DirectMedia Library (SDL), and Open Scene Graph (OSG) as the framework for our Portable CAVE application, which we named Immersive GIS. This application was designed with the ability to import the aforementioned data types, render them in stereo, and display them across the three screens of the CAVE. Immersive GIS also allows for the exaggeration of terrains, various texture overlays, simple navigation through a game pad, and pre-programmable flight paths through various terrains.

Introduction

CAVE stands for Cave Automatic Virtual Environment. It is a recursive acronym as well as a reference to Plato's "The Allegory of the Cave" from his famed work, *The Republic*, in which he uses the analogy of shadows dancing on his cave walls to define human perception and reality.[1] Scientists from the University of Illinois used this analogy in the creation of the first present-day CAVE. A traditional CAVE is a multi-person, immersive virtual reality environment. It consists of three walls and a floor traditionally, in which three-dimensional images are displayed stereoscopically. This environment is successful in allowing users to interact with data in ways never possible before. Traditional CAVEs use expensive, high-level technology and have an immobile structure. In past years, DEEVLOP created the hardware components for a low-cost, portable CAVE. The task this summer was to create the software application to render Earth Science visualizations to display them in stereo 3D in the CAVE, as well as improve upon the safety, reliability and portability of the hardware.

Background

In the summer of 2002, the DEVELOP program at NASA began work on an inexpensive, mobile, and user-friendly CAVE in order to gain a more effective method to display their Earth Science data visualizations at conferences and meetings. A traditional CAVE is comprised of expensive DLP active-stereo projectors, super computers, and can cost upwards of a million dollars. DEVELOP's CAVE has been implemented using commodity LCD projectors and a cluster of readily available Linux based gaming PC's. A traditional CAVE uses expensive DLP projectors, because a standard LCD projector cannot flip between the two stereo images fast enough. A method was developed to gain this stereo imaging without the use of expensive projectors by designing a shutter wheel system in which two projectors display the right and left eye image simultaneously. A shutter wheel system then controls which image is displayed at any given instant.

DEVELOP's CAVE project is in its third year. In the first year, the team showed the feasibility of using a cluster of computers to display stereo images on three screens. In the second year, the team designed and constructed the hardware, which included the collapsible screen frame, the mirror stands, the projector racks, and the shuttering system. In the third year, the software was designed to actively import and render data to be displayed in the CAVE. In that year, much of the hardware was improved on as well as a new, more efficient alignment system developed.

Hardware

Most of the hardware components were completed by past DEVELOP teams. Many unique methods were used to make the environment less costly and portable.
This mobile CAVE can be setup and aligned in less than two hours and disassembled in even less time. It fits comfortably into a fifteen-person van and taken to meetings and conferences all over the country (Figure 1).

Figure 1 CAVE packed in van

Active-Stereo Projection

Humans' perception of depth is the result of a complex neurological process that draws on everything from basic trigonometry to human experience. A CAVE exploits several of the brain's cues for depth perception to create the illusion of depth and three-dimensionality. The principal cue that a CAVE exploits is the disparity between the two eyes, which arises from their horizontal separation. Due to this disparity, the brain receives an image in stereo as two slightly different images as evidenced by basic trigonometry. In a CAVE, an image is displayed on each wall in stereo at a frequency greater than 80 Hz. This stereo projection used in conjunction with synchronized, liquid crystal shuttering glasses, tricks the user's brain into seeing depth, thus creating the desired immersive 3D environment. A CAVE system also exploits other cues, such as shading, texturing, and overlaying. These other cues are all created in software.

Cluster

In the summer of 2002, the students focused on building three display machines, each responsible for one wall of the CAVE, and a server. The open source Red Hat Linux operating system was loaded on each machine and they were connected through a high-speed network switch. Table 1 shows a summary of the computer's main hardware components.

	Server	Display Node
Graphics	NVidia GeoForce4 TI 4200	NVidia GeoForce4 TI 4600
Processor	Dual Athlon 1900+	Athlon 2000+
RAM	1.5 GB	1 GB
Hard Drive	160 GB RAID 0	120 GB RAID 0
Network	1000 Mb/s	100 Mb/s

Table 1 Cluster Specification [2]

Environment

The physical environment of the CAVE consists of the frame, screens, mirrors, and projectors. Square aluminum tubing was assembled to construct the frame of the CAVE. Each segment of the frame slides onto connectors in the corners (Figure 2). This arrangement allows the frame to easily be assembled and disassembled. The screen, which was sewn by a sail maker, then slides easily over the frame. Specially designed boxes house each pair of projectors to aid in alignment. These boxes were built in a cabinet shop and allow for pitch, roll, and yawl adjustments. The mirrors were held upright by simple mounts that allow for pitch adjustment. This entire system is easily transported, assembled, and aligned.

Figure 2 Corner

Improvements

The most obvious shortcoming of the hardware was the safety concerns and the lack of portability of the mirrors. Previously, the mirrors were held by a simple stand that did not provide much support or any protection. To amend this problem, we designed and built frames and new stands for the mirrors (Figure 3). Each frame was built with cabinet quality plywood to minimize warping and increase strength. The mirror then slid into a 2x4 frame, which had a half-inch groove cut into it. Padding was placed behind the mirrors for an added layer of protection. Padded covers were also made for the mirrors. These covers had wheels on the bottom to allow for easier transportation. To hold these frames, new stands were built that have the ability to collapse and be easily transported.

The next hardware improvement that was made was the simplification of the turn on/off process. Previously, a screwdriver was used to speed up and slow down the chopper wheels. Problems were encountered when starting and stopping at full speed. The incorporation of a capacitor into the oscillator circuit solved this problem. A 500 μF capacitor provided enough charge and discharge time to allow the motors to slowly accelerate and then decelerate with a simple switch.

The next hardware improvement was to rewire some of the circuitry that drives the hardware. This consisted of simple re-soldering and organization of the wires and connections. The components involved included the oscillator, stepper drivers, amplifiers, optical sensor, and emitters.

Figure 3
New Mirror Frame

Software

Throughout the first years in development, the efforts regarding the software aspect of the CAVE had for the most part lagged behind that of the development on the actual hardware, but with the hardware setup virtually complete this past summer, the focus of the team has been aimed primarily at adding the software component. There were of course many options to choose from on deciding the best path to completing the project each having its own advantages and disadvantages. Indeed much time was spent on researching these options before deciding what software needed to be programmed. At the end of this research a handful of important libraries were chosen and software was written to specifically handle the purpose of the cave. That is, the software that was to be created needed to be capable of the following: to render in real time to allow user interaction with the environment, to display vast terrains at varying levels of detail, to overlay geo-referenced raster and vector data sets over the terrain, and to provide a means of flying through the terrain using previously created flight paths.

Research

When this summer's team approached the task of designing active, running software for the Portable CAVE, we researched many possibilities. Past DEVELOP teams that could not fully complete tackled the task of designing the software used programs such as VRJuggler,[3] CAVELib,[4] and Vgeo[5] as their framework. We also explored Diverse[6] and writing the program from scratch. When choosing the direction to take, time and money needed to be greatly considered. Therefore we eliminated CAVELib from our list because it costs nearly $3,000. Also Diverse was a library that only worked with the graphics API, Open GL Performer, which would have to be purchased. VRJuggler was a very complex program, which would be quite time-consuming to learn. The main file types we need to support to show Earth Science visuals in the CAVE are DEMS, overlays, and Geospatial Feature Sets. Vgeo, a library looked very heavily at by past teams, was found to not support the data types that we needed. CAVELib and Vgeo also had licensing issues and could have possibly been a problem when it came time to take our CAVE off-center to show at other locations. Our choice then became using the Virtual Terrain Project (VTP),[7] Simple DirectMedia Library (SDL),[8] and Open Scene Graph (OSG)[9] as the framework for our Portable CAVE application.

The Virtual Terrain Project is a library based on the goal of creating tools for allowing a programmer to easily construct and display any part of the real world in a 3D digital format. Virtual Terrain Project allows a user to input elevation and texture overlay data rather easily. The SDL library allows a programmer to interrupt user input and utilize the input in other programs. For example, the game pad utilized by Immersive GIS is imported through the SDL library. Open Scene Graph is a library that can be utilized in the construction of a scene. It can render digital 3D objects. It aids in generating models and elevation visuals.

Immersive GIS

After reaching a decision to use the VTP, SDL, and OSG libraries as the basis for a C++ program for accomplishing the goals for the DEVELOP CAVE, some of the first problems were tackled. The most difficult obstacle to be encountered in programming the application was the complete lack of knowledge in using any of the base libraries, and learning them was made fairly difficult due to the poor documentation of some of them, especially VTP, for which we typically relied on using the source code for understanding its usage. Before introducing any complexities of writing the application to run in the CAVE, the application was created to run in a typical single-machine desktop environment; this made learning how to use the VTP library much simpler. The SDL library was used to construct the display window, manage user input, and to create the OpenGL graphics context in which OSG would render to in this simplified environment. Next, code was written to handle navigating with game pad devices, the rendering of terrain, and flying previously made flight paths.

These tasks were pretty much divided amongst individual team members. Using a program called Concurrent Versions System (CVS), each member would work on their specific part and update a centralized code repository with their changes. Managing the code development with this program worked out very well. For programming navigation of the scene with a game pad device was fairly straightforward; in fact the functionality provided by SDL for joystick devices was one of the reasons we opted to use it. The rendering of the terrain was extremely simplified through the use of the VTP library. The program was coded to load a terrain Extensible Markup Language (XML) script specified by a command line parameter. The main difficulty encountered here was preparing the data provided by the environmental science DEVELOP teams to something that could be loaded with the VTP library. Eventually the tools provided with the Geospatial Data Abstraction Library (GDAL)[10] library were discovered and used to transform the DEM and overlay data to something usable. Programming the navigator to fly an arbitrary flight path was one of the more challenging things to code. However, by adding some code, some functions of the OSG library were used to program the autonomous flight system.

Once the application was found to be working well, it was then extended to run in the CAVE environment. The application then needed the capability of synchronously running on three machines and displaying stereo graphics. Providing the stereo graphics was certainly the easier of the two tasks, as OSG had built-in stereo support. In fact, by simply setting specific environment variables recognizable to OSG, stereo graphics could be produced. Of course, there were problems with enabling stereo graphics through the environment variables that made it seem as if OSG was not working correctly, and it was not until the source code was delved into for a day or two that the source of the problem was discovered and fixed. The only other problems encountered in permitting stereo graphics were in coding VTP to use OSG's built-in stereo capabilities. Though learning the code took a while the solution only involved to small fixes to the library. The last problem involved creating an oblique projection matrix for the application for when it ran on a wall of the CAVE. The measurements for the frustum were calculated and put into a configuration script along with other window creating variables including resolution, and model view rotation – used to rotate the navigator to the right and left for creating the appropriate view for the side walls.

The solution to utilizing the distributed rendering system of the cave may not have been solved as nicely as it could have been, however a solution was found. While it may have been the goal of Open Producer, one of the libraries OSG is based on, to be able to visualize an OSG scene on any display setup including a CAVE, the library was not quite to that point in development. In the future, using this capability could simplify any coding and greatly increase the versatility of any software developed. The approach taken in the development of Immersive GIS was to use SDL_net, a small multi-platform library for providing network communication. A client/server model was used for the application. One machine would run the application and act as the server, while the other three machines would be run as clients. Simply put, the server application was programmed to continually broadcast the viewer's model viewing matrix to the client machines. In order to provide some degree of versatility for the application, the code was made to set up the client/server arrangement via a configuration file specified as a command-line argument.

Overall, the written from scratch program provided a great degree of performance and was well suited to its purpose, though undoubtedly future improvements could be made. Besides having used only open source libraries in creating Immersive GIS, they all were multi-platform. It would likely not be very difficult to compile and use the application on a different platform.

Running the CAVE

There are only four basic components in setting up the CAVE: the frame, the mirrors, the canvas, and the computers. The first step is assembling the frame, which is fairly intuitive, and positioning it in a room in such a way that the mirrors can then be placed a distance behind each wall. Diagram 1 best represents this configuration. Starting with the back wall, the canvas is then hung up around the frame and zipped. The vertical cables can afterwards be tightened to remove any wrinkles in the canvas. Alignment of the mirrors and projector boxes is the most difficult procedure in setting up the cave. Using a specially designed alignment string, the distances and angles between the wall, the mirror, and the projector box, can be set. Afterwards, to set up each wall requires time intensive tweaking of the aforementioned components positioning.

The computer system is relatively easy to set up. After placing the three display nodes alongside their corresponding projector boxes and plugging in all the cables, the Ethernet network cables are wired to a port on the switch. The server is set up out of the way of the CAVE and is connected to the gigabit port of the switch. Once all the machines are running, the application can be run. Though it is a bit awkward, the current way of running the Immersive GIS application is to initialize it on each individual machine through secure shell. Each program is run with the same command line option to use the cave configuration file. Which sets up the window and server/client role of the program on each specific machine. When executing the program on the display nodes, it will also be necessary to set a chain of OSG stereographic environment variables. Further documentation on how to use the program was created in the application directory.

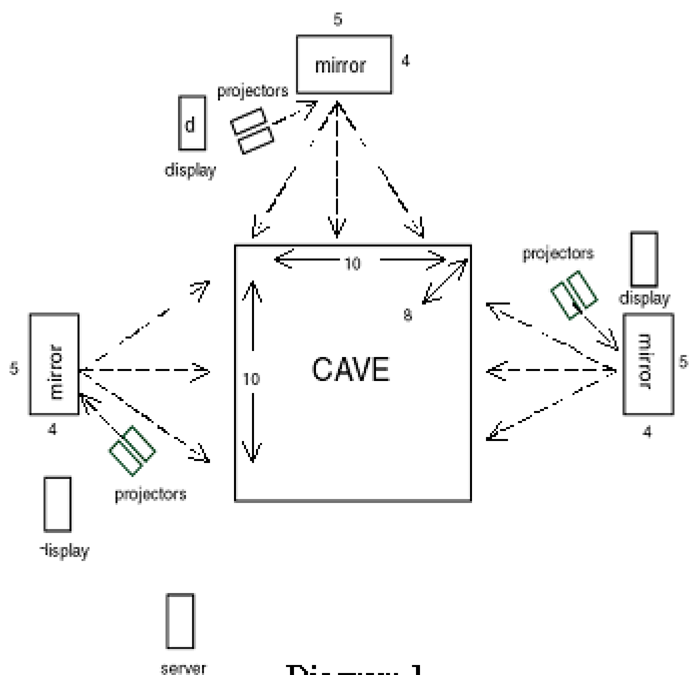

Diagram 1

LaRC Equipment/Facilities Used

The CAVE team received help from Dr. Sandridge, the head of one of Langley's two traditional CAVEs. He allowed us to view a space station application in his CAVE and offered us software suggestions in the beginning of our term. Mr. Gary Qualls, the NASA official responsible for the Arial Regional-scale Environmental Study of Mars, allowed us to view some of his three-dimensional animations of the mars airplane and helped us try to format it to view in the CAVE.

Further Research

Virtual reality, in its most general term is: "an artificial environment created with computer hardware and software and presented to the user in such a way that it appears and feels like a real environment."[11] Much research has been done on the uses and effects of virtual reality both physically and mentally. As with any research, both good and bad aspects have been discovered.

Benefits

Virtual reality can provide as an excellent training device for many industries. It can be used in its physical aspect for testing and research that when done without virtual reality would take much longer, more manpower, as well as possibly being a threat to the persons involved. The aviation industry has been turning to virtual reality for air traffic control purposes. The medical field is another prominent industry investigating using virtual reality simulators to create "virtual patients" for training and planning purposes. It will greatly reduce the cost and patient-risk during training.[12] Virtual Reality can also be of use to surgeons by allowing them to perform surgeries in remote locations. There is also a great deal of research being done on the mental benefits to virtual reality. In a recent study, a team of Seattle doctors and engineers are promoting virtual headgear as a method of reducing pain. In an ongoing study, it has been discovered that virtual reality helps ease pain even more than medication in some cases. This is due to the fact that the brain seems to put a lot of focus on pain. If the focus can be taken away by means of a different, complimentary environment, the patient seems to feel less pain.[13] An example given is a burn patient who needs to be scrubbed regularly for medical purposes, be engulfed in a snowy environment taking focus off of the burning sensation and making the patient feel a little chilly even.

Concerns

As with any new innovation, there are concerns that need to be considered when it comes to virtual reality especially when speaking in terms of a CAVE. A great deal of research still needs to be done on the effects of a person staying in a CAVE environment for great periods of time. Some people experience dizziness or nausea the minute they put the LCD glasses on and step into the CAVE, some after a long period of time, and some people feel no symptoms at all in the CAVE. Until further research is done, we chose to limit people's time in out portable CAVE to five minutes at a time. Another concern is seizures. Due to the flickering of the lights in the CAVE for the stereoscopically viewing, it could be a hazard to those who have a history of seizures especially photosensitive seizures.[14] To prevent any accidents or harm to a CAVE viewer we actively post signs warning the viewer of the flickering they will experience in the CAVE before they enter.

Conclusion/Results

The overall results of the project were very good. The software was successfully designed to display Earth Science visualizations in the CAVE. Also, hardware improvements made the CAVE more reliable, safer, and more "travel-friendly." As with any software program, enhancements can and will be made. Also, more can be done in the future to make the CAVE even more portable. Upon finding a new location with a higher ceiling for the Portable CAVE, a floor can be added to make it an exact replication of a traditional CAVE.

<u>References</u>

[1] Kenyon, Robert V. "THE **CAVE**™ AUTOMATIC VIRTUAL ENVIRONMENT: CHARACTERISTICS AND APPLICATIONS." *Human-Computer Interaction and Virtual Environments,* ed. Ahmed Noor Ph.D., NASA Conference Publication #3320, Pages 149-168, November, 1995.

[2] Miller, Samuel…

[3] VR Juggler. Ver. 2.0 for Linux. Open Source Library. Ames, Iowa: Iowa State University's Virtual Reality Applications Center, 2004.
Available: http://vrjuggler.org

[4] CAVELib. Ver. 3.0 for Linux. Licensed Library. Virginia Beach, Virginia: VRCO, 2003.
Available: http://www.vrco.com/products/cavelib/cavelib.html

[5] Virtual Global Explorer and Observatory. Ver. 2.0 for Linux**.** Licensed Library. Virginia Beach Virginia: VRCO, 2003.
Available: http://www.vrco.com/products/vgeo/vgeo.html

[6] Device Independent Virtual Environments- Reconfigurable, Scalable, Extensible. Ver. 2.3.2 for Linux. Open Source Library. Blacksburg, Virginia. University Visualization and Animation Group, 2003.
Available: http://diverse.sourceforge.net/

[7] Virtual Terrain Project. Ver. 2.0 for Linux. Open Source Library.

Available: http://www.vterrain.org/

[8] Simple DirectMedia Layer. Ver 1.2 for Linux. Cross-platform multimedia library. Sam Lantinga, Loki Entertainment Software.
Available: http://www.libsdl.org/index.php

[9] Open Scene Graph. Linux. Graphics API. Sourceforge.net, 2004.
Available: http://openscenegraph.sourceforge.net/index.html

[10] Geospatial Data Abstraction Library. Ver. 1.2.1 for Linux. Translator library for raster geospatial data formats. Doxygen, 2004.
Available: http://www.remotesensing.org/gdal/

[11] "Virtual Reality." *Webopedia*. Datamation. Retrieved 3 Aug 2004. <http://itmanagement.webopedia.com/TERM/v/virtual_reality.html>

[12] "Med students practice on 'virtual' patients." *eSchool News staff and wire service reports*. 4, Aug 2004. <http://www.eschoolnews.com/news/showStoryts.cfm?ArticleID=5212>

[13] Paulson, Tom. "VR devices trick the brain into ignoring pain." *Seattle Post-Intelligencer*. 1 Jul. 2004. <http://seattlepi.nwsource.com/health/180283_hcenter01.html>

[14] "Photosensitivity and Seizures." *Epilepsy Foundation Website.* 2002. <http://www.epilepsyfoundation.org/answerplace/Medical/seizures/precipitants/photosensitivity/photosensitivity.cfm>